AMENDMENTS TO THE CLAIMS

Kindly amend claims 1, 3, 4, 10, 24, 26, 32, 38, and 39 as shown in the following listing of claims. The listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims

1. (currently amended) A variable latency cache memory, comprising:

an input, for specifying a type of an instruction requesting to read data from the cache memory, wherein said type is one of a plurality of predetermined instruction types; and

a plurality of storage elements, coupled to said input, configured as a last-in-first-out (LIFO) memory, and configured to store data exclusively specified by push instructions, wherein each of said push instructions implicitly specifies a data memory address based on a value stored in a microprocessor stack pointer register rather than explicitly specified by the push instruction, and for providing said requested data in a first number of clock cycles if said input specifies a pop instruction type of said plurality of predetermined instruction types, and for providing said requested data in a second number of clock cycles if said input specifies a load instruction type of said plurality of predetermined instruction types, wherein said first and second number of clock cycles is different.

2. (canceled)

3. (currently amended) The cache memory of claim 1, further comprising:

a second plurality of storage elements, coupled to said first plurality of storage elements, for caching non-stack data~~, whereas said first plurality of storage elements is for caching stack data~~.

4. (currently amended) The cache memory of claim 3, wherein said second plurality of storage elements provides said requested data in a third number of clock cycles if said requested data is not present in said first plurality of storage elements and if said input specifies said load instruction type of said plurality of predetermined instruction types, wherein said second and third number of clock cycles is different.

5. (original) The cache memory of claim 4, wherein said third number of clock cycles is greater than said second number of clock cycles.

6-8. (canceled)

9. (original) The cache memory of claim 1, a computer program product comprising a computer usable medium having computer readable program code causes the

cache memory, wherein said computer program product is for use with a computing device.

10. (currently amended) A variable latency cache memory, comprising:

a plurality of storage elements, configured as a last-in-first-out (LIFO) memory, and configured to store data exclusively specified by push instructions, wherein each of said push instructions implicitly specifies a data memory address based on a value stored in a microprocessor stack pointer register rather than explicitly specified by the push instruction, having first and second subsets of said plurality of storage elements, said first subset for caching stack data more recently pushed than data cached in said second subset;

an input, for specifying a memory address of source data requested from the cache memory; and

at least one comparator, coupled to said input, for comparing said memory address with one or more memory addresses of said data cached in said first subset of storage elements, wherein if said memory address hits in said first subset based on said comparing, the cache memory provides said source data from said first subset in a first number of clock cycles, wherein if said memory address does not hit in said first subset but hits in said second subset based on said comparing, the cache memory provides said source data from said second subset in a second number of clock cycles, wherein said first and second number of clock cycles is different.

11. (previously presented) The cache memory of claim 10, wherein said first number of clock cycles is less than said second number of clock cycles.

12. (previously presented) The cache memory of claim 10, wherein said memory address comprises a virtual memory address.

13. (previously presented) The cache memory of claim 12, further comprising:

a second input, coupled to said plurality of storage elements, for specifying a physical memory address of said source data requested from the cache memory.

14. (previously presented) The cache memory of claim 13, further comprising:

a second at least one comparator, coupled to receive said second input, for comparing said physical memory address with one or more physical memory addresses of said data cached in said first subset of storage elements.

15. (previously presented) The cache memory of claim 14, further comprising:

an output, coupled to said first and second at least one comparator, for indicating an error condition if the cache memory provides said source data from said first subset based on said comparing said virtual memory address with said one or more memory addresses of said data cached in said first subset of

storage elements, but said second at least one comparator indicates said physical memory address does not match any of said one or more physical memory addresses of said data cached in said first subset of storage elements.

16. (original) The cache memory of claim 10, further comprising:

a second plurality of storage elements, coupled to said first plurality of storage elements, for caching non-stack data.

17. (original) The cache memory of claim 16, wherein said second plurality of storage elements provides said data in a third number of clock cycles if said address does not hit in said first plurality of storage elements, wherein said second and third number of clock cycles is different.

18. (original) The cache memory of claim 17, wherein said third number of clock cycles is greater than said second number of clock cycles.

19. (previously presented) The cache memory of claim 10, wherein if said address does not hit in said first subset based on said comparing, the cache memory provides said source data in said second number of clock cycles based on a physical address compare.

20. (original) The cache memory of claim 10, wherein said first subset comprises a top one of said plurality of storage elements.

21. (original) The cache memory of claim 10, wherein said first subset comprises a top two of said plurality of storage elements.

22. (original) The cache memory of claim 10, wherein said first subset comprises a top three of said plurality of storage elements.

23. (previously presented) The cache memory of claim 10, wherein said memory address comprises a source data address of a load instruction.

24. (currently amended) The cache memory of claim 10, wherein each of said plurality of storage elements is configured to store a cache line of data, wherein a cache line of data comprises at least 64 bytes of data ~~wherein a computer data signal embodied in a transmission medium comprising computer readable program code provides the cache memory~~.

25. (previously presented) The cache memory of claim 10, wherein a computer program product comprising a computer usable medium having computer readable program code causes the cache memory, wherein said computer program product is for use with a computing device.

26. (currently amended) A method for providing data from a cache memory with a variable latency, the method comprising:

storing stack data into the cache memory exclusively specified by push instructions, wherein each of the push instructions implicitly specifies a data memory address based on a value stored in a microprocessor stack

pointer register rather than explicitly specified by the push instruction ~~in a last-in-first-out manner~~;

providing load data from the cache memory in a first number of clock cycles if a virtual memory address of the load data hits in the cache memory, after said storing; and

providing the load data from the cache memory in a second number of clock cycles if the virtual memory address of the load data misses in the cache memory but a physical memory address of the load data hits in the cache memory, after said storing, wherein the first and second number of clock cycles is different.

27. (previously presented) The method of claim 26, further comprising:

determining whether the virtual memory address hits in a top subset of cache lines of the cache memory, wherein the top subset is less than all cache lines of the cache memory;

wherein said providing the load data from the cache memory in a first number of clock cycles if a virtual memory address of the load data hits in the cache memory is in response to said determining.

28. (original) The method of claim 27, wherein the top subset of cache lines of the cache memory comprises cache lines implicated by most recently pushed stack data.

29. (original) The method of claim 26, wherein the first number of clock cycles is less than the second number of clock cycles.

30. (previously presented) The method of claim 26, wherein said providing the load data from the cache memory in the first number of clock cycles if the virtual memory address of the load data hits in the cache memory is speculative subject to a subsequent determination that the physical memory address of the load data hits in the cache memory.

31. (previously presented) The method of claim 26, further comprising:

providing the load data from a non-stack cache memory in a third number of clock cycles if the virtual memory address and the physical memory address miss in the cache memory, wherein the first and third number of clock cycles is different.

32. (currently amended) A method for providing data from a last-in-first-out (LIFO) cache memory with a variable latency, the method comprising:

storing data into the LIFO cache memory exclusively specified by push instructions, wherein each of the push instructions implicitly specifies a data memory address based on a value stored in a microprocessor stack pointer register rather than explicitly specified by the push instruction;

determining whether a request for data from the LIFO cache memory is in response to a pop or load instruction, prior to said storing;

providing the requested data from the LIFO cache memory in a first number of clock cycles if the request is in response to a pop instruction; and

providing the requested data from the LIFO cache memory in a second number of clock cycles if the request is in response to a load instruction, wherein the first and second number of clock cycles is different.

33. (original) The method of claim 32, wherein the first number of clock cycles is less than the second number of clock cycles.

34. (original) The method of claim 32, wherein said providing the data in the first number of clock cycles if the request is in response to a pop instruction is speculative subject to a subsequent determination that a source address of the data hits in the cache memory.

35. (original) The method of claim 32, wherein a load instruction comprises an instruction explicitly specifying a source address of the data.

36. (original) The method of claim 32, wherein a pop instruction comprises an instruction inherently specifying a source address of the data.

37. (original) The method of claim 36, wherein the pop instruction inherently specifies the source address of the data relative to a stack pointer value.

38. (currently amended) A computer program product embodied on a computer-readable medium, comprising:

computer-readable program code for providing a variable latency cache memory, said program code comprising:

first program code for providing an input, for specifying a type of an instruction requesting to read data from the cache memory, wherein said type is one of a plurality of predetermined instruction types; and

second program code for providing a plurality of storage elements, configured as a last-in-first-out (LIFO) memory, coupled to said input, and configured to store data exclusively specified by push instructions, wherein each of said push instructions implicitly specifies a data memory address based on a value stored in a microprocessor stack pointer register rather than explicitly specified by the push instruction, for providing said requested data in a first number of clock cycles if said input specifies a pop instruction type of said plurality of predetermined instruction types, and for providing said requested data in a second number of clock cycles if said input specifies a load instruction type of said plurality of predetermined instruction types, wherein said first and second number of clock cycles is different.

39. (currently amended) A computer program product embodied on a computer-readable medium, comprising:

computer-readable program code for providing a variable latency cache memory, said program code comprising:

first program code for providing a plurality of storage elements, configured as a last-in-first-out (LIFO) memory, and configured to store data exclusively specified by push instructions, wherein each of said push instructions implicitly specifies a data memory address based on a value stored in a microprocessor stack pointer register rather than explicitly specified by the push instruction, having first and second subsets of said plurality of storage elements, said first subset for caching stack data more recently pushed than data cached in said second subset;

second program code for providing an input, for specifying a memory address of source data requested from the cache memory; and

third program code for providing at least one comparator, coupled to said input, for comparing said memory address with one or more memory addresses of said data cached in said first subset of storage elements, wherein if said memory address hits in said first subset based on said comparing, the cache memory provides said source data from said first subset in a first number of clock cycles, wherein if said memory address does not hit in said first subset but hits in said second subset based on said comparing, the cache memory provides said source data from said second subset in a second number of clock cycles, wherein said first and second number of clock cycles is different.